WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, April 26, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT; PLUR=NO; OP=ADJ	
☐	L51	l24 and L49	0
☐	L50	l28 same L49	0
☐	L49	busy with (l12 or l13)	118
☐	L48	L45 and l24	0
☐	L47	L45 and l28	1
☐	L46	L45 same l28	0
☐	L45	L44 same (l12 or l13)	47
☐	L44	(forward Compatibil$3) or (backward compatibil$3)	1895
☐	L43	L42 same (l12 or l13)	7
☐	L42	(new or newer) with (coprocessor$1 or co-processor$1)	297
☐	L41	L40 not l32	18
☐	L40	(l12 or l13) and (l38 or l39)	18
☐	L39	old with (coprocessor$1 or co-processor$1)	26
☐	L38	older with (coprocessor$1 or co-processor$1)	8
☐	L37	obsolete with (coprocessor$1 or co-processor$1)	0
☐	L36	obsolete adj (coprocessor$1 or co-processor$1)	0
☐	L35	older adj (coprocessor$1 or co-processor$1)	0
☐	L34	old adj (coprocessor$1 or co-processor$1)	0
☐	L33	L32 not l30	6
☐	L32	l31 and (l25 or l26)	9
☐	L31	(l12 or l13) same l28	532
☐	L30	l15 and (l25 or l26)	3
☐	L29	L27 and L28	4
☐	L28	coprocessor$1 or co-processor$1	8047
☐	L27	L24 and L25	8
☐	L26	(703/26).ccls.	281
☐	L25	(712/227).ccls.	448
☐	L24	(712/34).ccls.	217
☐	L23	condition code$1 or predicate$1	18466
☐	L22	L6 and L20	81
☐	L21	L6 same L20	2

☐	L20	(co-processor$1 or coprocessor$1)	8047
☐	L19	L6 and L14	2
☐	L18	L17 not L15	51
☐	L17	(L12 or L13) and L16	59
☐	L16	(712/35).ccls.	185
☐	L15	(L12 or L13) and L14	71
☐	L14	(712/34).ccls.	217
☐	L13	(interface$1) with program$5	80590
☐	L12	(interface$1) with configur$4	31675
☐	L11	(L8 or L9) with configur$4	6
☐	L10	(L8 or L9) with program$5	37
☐	L9	co-processor interface	79
☐	L8	coprocessor interface	226
☐	L7	coprocessor with functional units	40
☐	L6	data with L5	2654
☐	L5	L4 adj L3	24603
☐	L4	out	2555749
☐	L3	order	2415057
☐	L2	out of order	0
☐	L1	"out of order"	0

END OF SEARCH HISTORY

First Hit

Generate Collection Print

L30: Entry 1 of 3 File: PGPB Jan 31, 2002

DOCUMENT-IDENTIFIER: US 20020013892 A1
TITLE: EMULATION COPROCESSOR

Current US Classification, US Primary Class/Subclass:
712/227

Current US Classification, US Secondary Class/Subclass:
712/34

Summary of Invention Paragraph:
[0005] As the above illustrates, a large diverse set of uses for computer systems have been developed. Generally, these uses are supported by a variety of application programs designed to execute under an operating system provided for the computer system. The operating system provides an interface between the application programs and the computer system hardware. Each computer system may have a variety of differences in hardware configuration (e.g. amount of memory, number and type of input/output (I/O) devices, etc.). The operating system insulates the application program from the hardware differences. Accordingly, the application program may often times be designed without regard for the exact hardware configuration upon which the application program is to execute. Additionally, the operating system provides a variety of low level services which many different types of application programs may need, allowing the application programs to rely on the operating system services instead of programming these services internal to the application program. Generally, the operating system provides scheduling of tasks (e.g. different application programs which may be operating concurrently), management and allocation of system resources such as I/O devices and memory, error handling (e.g. an application program operating erroneously), etc. Examples of operating systems are the Windows operating system (including Windows 95 and Windows NT), UNIX, DOS, and MAC-OS, among others. Conversely, an application program provides specific user functionality to accomplish a specific user task. Word processors, spreadsheets, graphics design programs, inventory management programs, etc. are examples of application programs.

Detail Description Paragraph:
[0054] Host processor core 48 is configured to fetch instructions from instruction cache 44 and to execute those instructions. The instructions may comprise a portion of a host application program, or may comprise a portion of the operating system employed by computer system 5. One particular portion of the operating system is used to create processes, including initiating a foreign application program. If, during execution of the create process portion of the operating system, a foreign application program is detected as being initiated, host processor core 48 communicates via command interface 54 with emulation coprocessor core 50. Host processor core 48 establishes a context within emulation coprocessor core 50 corresponding to the foreign application program being initiated. Included in the context is an initial program counter address, from which the first instruction in the foreign application program is to be fetched. Once the context is established, host processor core 48 provides a command to emulation coprocessor core 50 to begin execution. Emulation coprocessor core 50 begins fetching instructions at the program counter address, and executes the instructions according to the foreign instruction set architecture. As used herein, the term "context" refers to values

which are particular to a process. The context generally includes the memory pages allocated to the process, as well as register values.

Detail Description Paragraph:
[0058] Host processor core 48 and emulation coprocessor core 50 share MMU 42 in this embodiment as well. MMU 42 is configured to provide translations from the virtual addresses generated via execution of instructions in host processor core 48 and emulation coprocessor core 50 to physical addresses which bus interface unit 40 may use to read main memory 14 or L2 cache 38. Instruction cache 44 and data cache 46 may also store instructions and data according to physical addresses, in which case MMU 42 may be accessed in parallel with instruction cache 44 and data cache 46.

Detail Description Paragraph:
[0065] Host process 80 further includes emulation interface code 92 which may be used to communicate between the host processor and the emulation coprocessor. Accordingly, operating system thunks 86 may lead to invocation of emulation interface code 92 to pass messages between the host processor and emulation coprocessor. Furthermore, the host processor may be configured to request context information from the emulation coprocessor using emulation interface code 92. While the operating system routines being called by foreign application program 82 and corresponding operating system routines provided by operating system 84 provide the same function, the calling conventions (i.e. the manner in which parameters are passed between the application and the operating system routine) are different because the instruction set architectures are different. For example, the number and type of registers differ, and therefore the ability to pass parameters within the registers (as opposed to memory locations) differs. Accordingly, emulation interface code 92 may request the context values which are the parameters for the call, and may place the parameters in the corresponding registers on the host processor. The operating system call may then be performed by the host processor. Subsequently. the results of the operating system routine may be placed into the emulation coprocessor by reversing the conversion of calling conventions.

Detail Description Paragraph:
[0068] Upon receiving a command from a user to initiate an application program, the operating system creates a process in which the application program executes. The operating system examines the file format of the application program to determine what type of code is included in the application program (step 100). For an embodiment employing the Windows NT operating system, for example, the portable execution format includes an indication of which instruction set architecture the application program is coded for. The portable execution format is defined as part of application programming interface defined by Windows NT.

Detail Description Paragraph:
[0069] If the application program is determined to be coded according to the host instruction set architecture (decision block 102), the operating system establishes the process to as a normal host process and the application program is executed by the host processor (step 104). On the other hand, if the application program is determined not to be coded according to the host instruction set architecture, the operating system determines if the application program is coded according to a foreign instruction set architecture which is executable by an emulation coprocessor within the computer system (decision block 106). If the foreign instruction set architecture is executable by the emulation coprocessor, the operating system invokes the emulation coprocessor interface code in order to initiate the foreign application program upon the emulation coprocessor (step 108). If the foreign instruction set architecture is not executable by the emulation coprocessor, the operating system displays a message to the user indicating that the application is unsupported (step 110). The application program is not started in this case. Alternatively, software emulation or binary translation of the application may be provided at step 110 if desired. For example, a scheme similar

to Digital Equipment Corporation's FX!32 product or Microsoft's Wx86 product may be employed.

Detail Description Paragraph:
[0085] According to one particular embodiment, processor 10 as shown in FIG. 7 comprises three separate semiconductor chips attached to a printed circuit board. The printed circuit board may include an edge connector and be encapsulated for inclusion in computer system 5. For example, processor 10 may be designed in accordance with any of the slot 1, slot A, or slot 2000 specifications developed by Intel and Advanced Micro Devices. One chip embodies emulation coprocessor 150. A second chip embodies host processor 152, and a third chip embodies interface logic 154. For example, emulation coprocessor 150 and host processor 152 may be custom designed semiconductor chips and interface logic unit 154 may be an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), etc. Other organizations are possible and contemplated, including realizing interface logic unit 154 as a custom semiconductor chip as well.

Detail Description Paragraph:
[0103] In one particular embodiment, computer system 5 employs the Windows NT operating system for the Alpha instruction set architecture and host processor 150 employs the Alpha instruction set architecture. Furthermore, the Windows NT operating system employed by computer system 5 includes the Wx86 emulation extensions. However, the code for emulating the x86 processor is replaced by the emulation interface code described above. The driver for emulation coprocessor card 22C provides the page locking and unlocking functionality in response to lock and unlock requests from the executive. More particularly, the executive requests a locked page for either code or data. The driver, in response to the request, uses the Windows NT memory manager application programming interface (API) calls to lock the page (i.e. prevent swapping the page to disk to allow a different virtual page to be assigned to that physical page). Subsequently, the executive may determine that the page is no longer needed for application program execution and may send an unlock message. In response, the driver uses the Windows NT memory manager API to unlock the page. Additionally, the driver is responsible for initializing the card within the operating system and mapping the memory upon the card.

Detail Description Paragraph:
[0106] Turning next to FIG. 13 a block diagram of one embodiment of emulation coprocessor card 22C is shown. Other embodiments are possible and contemplated. As shown in FIG. 13, emulation coprocessor card 22C includes a PCI interface 190, the emulation coprocessor 150, and a memory 194. PCI interface 190 is coupled to PCI bus 24, memory 194, and emulation coprocessor 150. Emulation coprocessor 150 is further coupled to memory 194. Memory 194 includes storage for the executive program 196 and for the command queues 198 used to pass command messages between executive program 196 and the driver for emulation coprocessor card 22C as well as emulation interface code 92. In other words, command queues 198 may comprise command interface 54. It is noted that, while instructions and data are preferably accessed from main memory 14 directly by emulation coprocessor 150, alternative embodiments may store instructions and data transferred from pages in main memory 14 in memory 194 as well.

First Hit Fwd Refs

Generate Collection Print

L33: Entry 1 of 6 File: USPT Nov 12, 2002

US-PAT-NO: 6480952
DOCUMENT-IDENTIFIER: US 6480952 B2

TITLE: Emulation coprocessor

DATE-ISSUED: November 12, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gorishek, IV; Frank J.	Austin	TX		
Boswell, Jr.; Charles R.	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Advanced Micro Devices, Inc.	Sunnyvale	CA			02

APPL-NO: 09/ 085187 [PALM]
DATE FILED: May 26, 1998

INT-CL: [07] G06 F 9/455, G06 F 9/54, G06 F 13/38

US-CL-ISSUED: 712/227; 712/36, 712/209, 703/26, 703/27, 709/319, 709/328
US-CL-CURRENT: 712/227; 703/26, 703/27, 712/209, 712/36, 719/319, 719/328

FIELD-OF-SEARCH: 712/41, 712/208, 712/233, 712/23, 712/36, 712/31, 712/209, 712/212, 712/34, 712/227, 711/119, 711/121-143, 711/146, 709/208, 709/330, 709/329, 709/328, 709/319, 710/127, 710/128, 710/129, 710/101, 710/137, 710/130, 710/5, 710/55, 710/131, 710/150, 703/23, 703/26, 703/27, 703/29, 361/183, 717/5

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4799150	January 1989	Bui	710/130
☐	4954949	September 1990	Rubin	710/101
☐	5077657	December 1991	Cooper et al.	703/26
☐	5574927	November 1996	Scantlin	712/41
☐	5638525	June 1997	Hammond et al.	712/209
☐	5659709	August 1997	Quach	711/146

☐	5666519	September 1997	Hayden	703/23
☐	5717898	February 1998	Kagan et al.	711/145
☐	5764934	June 1998	Fisch et al.	710/129
☐	5802373	September 1998	Yates et al.	717/5
☐	5819105	October 1998	Moriarty et al.	710/5
☐	5802577	June 1999	Bhat et al.	711/146
☐	5909559	June 1999	So	710/127
☐	5923892	July 1999	Levy	712/31
☐	5925123	July 1999	Tremblay et al.	712/212
☐	6026238	February 2000	Bond et al.	709/528
☐	6275938	August 2001	Bond et al.	713/200
☐	6308255	October 2001	Gorisheck, IV, et al.	712/209

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 230 353	July 1987	EP	
0 455 345	November 1991	EP	
0 617 376	September 1994	EP	
0 671 685	February 1995	EP	
0 817 096	January 1998	EP	
2 278 214	November 1994	GB	
93/16437	August 1993	WO	

OTHER PUBLICATIONS

BYTE Magazine, "Why the 615 Matters", Linley Gwennap, 1995, pp. 198-200.
BYTE Magazine, "An Alpha in PC Clothing", Tom Thompson, Feb. 1996, pp. 1-6.
BYTE Magazine, "Alpha Learns to Do Windows", Selinda Chiquoine, Oct. 1996, pp. 1-3.

Microsoft Windows NT Workstation, "Windows(r) x86 Technology Preview", Microsoft Corporation, 1997, pp. 1.
AMD, "AMD-K6 Processor Data Sheet", Chpt.1 & 2, Mar. 1998, pp. 1-20.
Digital Equipment Corp., Maynard Massachusetts, "Digital Semiconductor 21172 Core Logic Chipset", Technical Reference Manual, Apr. 1996, pp. 1-1 to 1-7.
Readwx86, "Win32 x86 Emulation on Risc (Wx86)", pp.2.
AMD Tech Law "Intel Reveals how Merced with be x86-Compatible", Alexander Wolfe, Santa Clara, CA, May 8, 1998, pp. 2.
AMD, "AMD5.sub.k 86.TM. Processor", Technical Reference Manual, Chpts. 1 & 2, pp. 1-1 to 1-3 and 2-1 to 2-12.
"Press Release", Undated: Jun. 25, 1997, downloaded from: digital.com/info/PR00UM and www.zdnet.com/cshopper/content/9704/cshp0039.html on Dec. 11, 1997, (7 sheets).

Digital, "Digital FX!32", Updated: Sep. 26, 1997, downloaded from digital.com/semiconductor/amt/fx32/fx-white.html on Jan. 26, 1998, pp. 1-4.
Orange Micro, Inc., "Product Information", downloaded from orangemicro.com.page2.html on Aug. 4, 1998, (6 sheets).

ART-UNIT: 2183

PRIMARY-EXAMINER: Pan; Daniel H.

ATTY-AGENT-FIRM: Conley, Rose & Tayon, PC Merkel; Lawrence J.

ABSTRACT:

A computer system employing a host processor and an emulation coprocessor. The host processor includes hardware configured to execute instructions defined by a host instruction set architecture, while the emulation coprocessor includes hardware configured to execute instructions defined by a different instruction set architecture from the host instruction set architecture ("the foreign instruction set architecture"). The host processor core executes operating system code as well as application programs which are coded in the host instruction set architecture. Upon initiation of a foreign application program, the host processor core communicates with the emulation coprocessor core to cause the emulation coprocessor core to execute the foreign application program. Accordingly, application programs coded according to the foreign instruction set architecture can be executed directly in hardware. The computer system may be characterized as a heterogeneous multiprocessing system. While the emulation coprocessor is executing the foreign application program, the host processor may execute operating system routines unrelated to the foreign application program or may execute a host application program.

20 Claims, 15 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L33: Entry 1 of 6 File: USPT Nov 12, 2002

DOCUMENT-IDENTIFIER: US 6480952 B2
TITLE: Emulation coprocessor

Detailed Description Text (18):
Host processor core 48 is configured to fetch instructions from instruction cache 44 and to execute those instructions. The instructions may comprise a portion of a host application program, or may comprise a portion of the operating system employed by computer system 5. One particular portion of the operating system is used to create processes, including initiating a foreign application program. If, during execution of the create process portion of the operating system, a foreign application program is detected as being initiated, host processor core 48 communicates via command interface 54 with emulation coprocessor core 50. Host processor core 48 establishes a context within emulation coprocessor core 50 corresponding to the foreign application program being initiated. Included in the context is an initial program counter address, from which the first instruction in the foreign application program is to be fetched. Once the context is established, host processor core 48 provides a command to emulation coprocessor core 50 to begin execution. Emulation coprocessor core 50 begins fetching instructions at the program counter address, and executes the instructions according to the foreign instruction set architecture. As used herein, the term "context" refers to values which are particular to a process. The context generally includes the memory pages allocated to the process, as well as register values.

Detailed Description Text (22):
Host processor core 48 and emulation coprocessor core 50 share MMU 42 in this embodiment as well. MMU 42 is configured to provide translations from the virtual addresses generated via execution of instructions in host processor core 48 and emulation coprocessor core 50 to physical addresses which bus interface unit 40 may use to read main memory 14 or L2 cache 38. Instruction cache 44 and data cache 46 may also store instructions and data according to physical addresses, in which case MMU 42 may be accessed in parallel with instruction cache 44 and data cache 46.

Detailed Description Text (29):
Host process 80 further includes emulation interface code 92 which may be used to communicate between the host processor and the emulation coprocessor. Accordingly, operating system thunks 86 may lead to invocation of emulation interface code 92 to pass messages between the host processor and emulation coprocessor. Furthermore, the host processor may be configured to request context information from the emulation coprocessor using emulation interface code 92. While the operating system routines being called by foreign application program 82 and corresponding operating system routines provided by operating system 84 provide the same function, the calling conventions (i.e. the manner in which parameters are passed between the application and the operating system routine) are different because the instruction set architectures are different. For example, the number and type of registers differ, and therefore the ability to pass parameters within the registers (as opposed to memory locations) differs. Accordingly, emulation interface code 92 may request the context values which are the parameters for the call, and may place the parameters in the corresponding registers on the host processor. The operating system call may then be performed by the host processor. Subsequently, the results of the operating system routine may be placed into the emulation coprocessor by

reversing the conversion of calling conventions.

Detailed Description Text (33):
If the application program is determined to be coded according to the host instruction set architecture (decision block 102), the operating system establishes the process to as a normal host process and the application program is executed by the host processor (step 104). On the other hand, if the application program is determined not to be coded according to the host instruction set architecture, the operating system determines if the application program is coded according to a foreign instruction set architecture which is executable by an emulation coprocessor within the computer system (decision block 106). If the foreign instruction set architecture is executable by the emulation coprocessor, the operating system invokes the emulation coprocessor interface code in order to initiate the foreign application program upon the emulation coprocessor (step 108). If the foreign instruction set architecture is not executable by the emulation coprocessor, the operating system displays a message to the user indicating that the application is unsupported (step 110). The application program is not started in this case. Alternatively, software emulation or binary translation of the application may be provided at step 110 if desired. For example, a scheme similar to Digital Equipment Corporation's FX!32 product or Microsoft's Wx86 product may be employed.

Detailed Description Text (49):
According to one particular embodiment, processor 10 as shown in FIG. 7 comprises three separate semiconductor chips attached to a printed circuit board. The printed circuit board may include an edge connector and be encapsulated for inclusion in computer system 5. For example, processor 10 may be designed in accordance with any of the slot 1, slot A, or slot 2000 specifications developed by Intel and Advanced Micro Devices. One chip embodies emulation coprocessor 150. A second chip embodies host processor 152, and a third chip embodies interface logic 154. For example, emulation coprocessor 150 and host processor 152 may be custom designed semiconductor chips and interface logic unit 154 may be an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), etc. Other organizations are possible and contemplated, including realizing interface logic unit 154 as a custom semiconductor chip as well.

Detailed Description Text (67):
In one particular embodiment, computer system 5 employs the Windows NT operating system for the Alpha instruction set architecture and host processor 150 employs the Alpha instruction set architecture. Furthermore, the Windows NT operating system employed by computer system 5 includes the Wx86 emulation extensions. However, the code for emulating the x86 processor is replaced by the emulation interface code described above. The driver for emulation coprocessor card 22C provides the page locking and unlocking functionality in response to lock and unlock requests from the executive. More particularly, the executive requests a locked page for either code or data. The driver, in response to the request, uses the Windows NT memory manager application programming interface (API) calls to lock the page (i.e. prevent swapping the page to disk to allow a different virtual page to be assigned to that physical page). Subsequently, the executive may determine that the page is no longer needed for application program execution and may send an unlock message. In response, the driver uses the Windows NT memory manager API to unlock the page. Additionally, the driver is responsible for initializing the card within the operating system and mapping the memory upon the card.

Detailed Description Text (70):
Turning next to FIG. 13 a block diagram of one embodiment of emulation coprocessor card 22C is shown. Other embodiments are possible and contemplated. As shown in FIG. 13, emulation coprocessor card 22C includes a PCI interface 190, the emulation coprocessor 150, and a memory 194. PCI interface 190 is coupled to PCI bus 24, memory 194, and emulation coprocessor 150. Emulation coprocessor 150 is further

coupled to memory 194. Memory 194 includes storage for the executive program 196 and for the command queues 198 used to pass command messages between executive program 196 and the driver for emulation coprocessor card 22C as well as emulation interface code 92. In other words, command queues 198 may comprise command interface 54. It is noted that, while instructions and data are preferably accessed from main memory 14 directly by emulation coprocessor 150, alternative embodiments may store instructions and data transferred from pages in main memory 14 in memory 194 as well.

Current US Original Classification (1):
712/227

Current US Cross Reference Classification (1):
703/26

First Hit Fwd Refs

Generate Collection Print

L33: Entry 3 of 6 File: USPT Oct 23, 2001

US-PAT-NO: 6308255
DOCUMENT-IDENTIFIER: US 6308255 B1

TITLE: Symmetrical multiprocessing bus and chipset used for coprocessor support allowing non-native code to run in a system

DATE-ISSUED: October 23, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gorishek, IV; Frank J.	Austin	TX		
Boswell, Jr.; Charles R.	Austin	TX		
Smith; David W.	Cedar Park	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Advanced Micro Devices, Inc.	Sunnyvale	CA			02

APPL-NO: 09/ 085188 [PALM]
DATE FILED: May 26, 1998

INT-CL: [07] G06 F 9/455, G06 F 9/44

US-CL-ISSUED: 712/209; 712/227, 710/102, 710/103, 710/104, 710/62, 710/8, 710/129, 710/126
US-CL-CURRENT: 712/209; 710/104, 710/301, 710/302, 710/305, 710/62, 710/8, 712/227

FIELD-OF-SEARCH: 712/1, 712/28, 712/32, 712/23, 712/33, 712/24, 712/34, 712/35, 712/41, 712/43, 712/209, 712/227, 712/229, 712/38, 709/213, 709/229, 709/250, 710/1, 710/2, 710/62, 710/72, 710/101, 710/102, 710/240, 711/146, 703/26, 703/27, 717/5

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4799150	January 1989	Bui	710/130
☐	4954949	September 1990	Rubin	710/101
☐	5077657	December 1991	Cooper et al.	395/500
☐	5574927	November 1996	Scantlin	712/41

☐	5638525	June 1997	Hammond et al.	712/209
☐	5659709	August 1997	Quach	711/146
☐	5666519	September 1997	Hayden	712/233
☐	5717898	February 1998	Kagan et al.	711/145
☐	5764934	June 1998	Fisch et al.	710/129
☐	5802373	September 1998	Yates et al.	717/5
☐	5802577	June 1999	Bhat et al.	711/146
☐	5819105	October 1998	Moriarty et al.	395/825
☐	5909559	June 1999	So	395/307
☐	5923892	July 1999	Levy	712/31
☐	5925123	July 1999	Tremblay et al.	712/212

FOREIGN PATENT DOCUMENTS

FOREIGN-PAT-NO	PUBN-DATE	COUNTRY	US-CL
0 230 353	July 1987	EP	
0 455 345	November 1991	EP	
0 617 376	September 1994	EP	
0 671 685 A2	February 1995	EP	
0 817 096	January 1998	EP	
2 278 214	November 1994	GB	

OTHER PUBLICATIONS

BYTE Magazine, "Why the 615 Matters", Linley Gwennap, 1995, pp. 198-200.
BYTE Magazine, "An Alpha in PC Clothing", Tom Thompson, Feb. 1996, pp. 1-6.
BYTE Magazine, "Alpha Learns to do Windows", Selinda Chiquoine, Oct. 1996, pp. 1-3.

Microsoft Windows NT Workstation, "Windows (r) x86 Technology Preview", Microsoft Corporation, 1997, pp. 1.
AMD, "Amd-K6 Processor Data Sheet", Chpts. 1 & 2, Mar. 1998, pp. 1-20.
Digital Equipment Corp., Maynard Massachusetts, "Digital Semiconductor 21172 Core Logic Chipset", Technical Reference Manual, Apr. 1996, pp. 1-1 to 1-7.
Readwx86, "Win32 .times.86 Emulation on Risc (W.times.86)", pp. 2.
AMD Tech Law, "Intel Reveals How Merced Will be .times.86 -Compatible", Alexander Wolfe, Santa Clara, CA, May 8, 1998, pp. 2.
AMD, "AMD5K 86TM Processor", Technical Reference Manual, Chpts. 1 & 2, pp. 1-1 to 1-3 and 2-1 to 2-12.
"Press Release", Updated: Jun. 25, 1997, downloaded from:www.digital.com/info/PR00UM and www.zdnet.com/cshopper/content/9704/cshp0039.html on Dec. 11, 1997, (7 sheets).
Digital, "Digital FX!32", Updated Sep. 26, 1997, downloaded from www.digital.com/semiconductor/amt/fx32/fx-white.html on Jan. 26, 1998, pp. 1-4.
Orange Micro, Inc., "Product Information", downloaded from www.orangemicro.com.page2.html on Aug. 4, 1998, (6 sheets).
International Search Report for Application No. PCT/US99/01457 mailed Jun. 14, 1999.

ART-UNIT: 213

PRIMARY-EXAMINER: Pan; Daniel H.

ATTY-AGENT-FIRM: Conley, Rose & Tayon, PC Merkel; Lawrence J.

ABSTRACT:

A computer system includes a host processor and an emulation coprocessor. The host processor includes hardware configured to execute instructions defined by a host instruction set architecture, while the emulation coprocessor includes hardware configured to execute instructions defined by a different instruction set architecture from the host instruction set architecture ("the foreign instruction set architecture"). According to one embodiment, the host processor executes operating system code as well as application programs which are coded in the host instruction set architecture. Upon initiation of a foreign application program, the host processor communicates with the emulation coprocessor to cause the emulation coprocessor to execute the foreign application program. The computer system also includes a bus bridge coupled to the host processor and the emulation coprocessor. The bus bridge provides access to main memory both for the host processor and the emulation coprocessor, and provides for coherency between the host processor and emulation coprocessor. Preferably in one particular embodiment, the bus bridge may be a bus bridge designed for a symmetric multiprocessing system including multiple host processors. By providing an emulation coprocessor having a bus interface which is electrically and logically identical to the bus interface provided by the host processor, the emulation coprocessor may be inserted into a processor slot within a symmetric multiprocessing system to form a computer system which employs high performance hardware support for a foreign instruction set architecture. The host processor may control the emulation coprocessor via software, allowing the coprocessor system to be realized without additional hardware.

15 Claims, 10 Drawing figures

L33: Entry 3 of 6 File: USPT Oct 23, 2001

DOCUMENT-IDENTIFIER: US 6308255 B1
TITLE: Symmetrical multiprocessing bus and chipset used for coprocessor support allowing non-native code to run in a system

Detailed Description Text (38):
Host process 80 further includes emulation interface code 92 which may be used to communicate between the host processor and the emulation coprocessor. Accordingly, operating system thunks 86 may lead to invocation of emulation interface code 92 to pass messages between the host processor and emulation coprocessor. Furthermore, the host processor may be configured to request context information from the emulation coprocessor using emulation interface code 92. While the operating system routines being called by foreign application program 82 and corresponding operating system routines provided by operating system 84 provide the same function, the calling conventions (i.e. the manner in which parameters are passed between the application and the operating system routine) are different because the instruction set architectures are different. For example, the number and type of registers differ, and therefore the ability to pass parameters within the registers (as opposed to memory locations) differs. Accordingly, emulation interface code 92 may request the context values which are the parameters for the call, and may place the parameters in the corresponding registers on the host processor. The operating system call may then be performed by the host processor. Subsequently, the results of the operating system routine may be placed into the emulation coprocessor by reversing the conversion of calling conventions.

Detailed Description Text (42):
If the application program is determined to be coded according to the host instruction set architecture (decision block 102), the operating system establishes the process as a normal host process and the application program is executed by the host processor (step 104). On the other hand, if the application program is determined not to be coded according to the host instruction set architecture, the operating system determines if the application program is coded according to a foreign instruction set architecture which is executable by an emulation coprocessor within the computer system (decision block 106). If the foreign instruction set architecture is executable by the emulation coprocessor, the operating system invokes the emulation coprocessor interface code in order to initiate the foreign application program upon the emulation coprocessor (step 108). If the foreign instruction set architecture is not executable by the emulation coprocessor, the operating system displays a message to the user indicating that the application is unsupported (step 110). The application program is not started in this case. Alternatively, software emulation or binary translation of the application may be provided at step 110 if desired. For example, a scheme similar to Digital Equipment Corporation's FX!32 product or Microsoft's Wx86 product may be employed.

Current US Cross Reference Classification (7):
712/227

First Hit Fwd Refs

Generate Collection Print

L33: Entry 4 of 6 File: USPT Feb 20, 2001

US-PAT-NO: 6192491
DOCUMENT-IDENTIFIER: US 6192491 B1

TITLE: Data processor with CRC instruction set extension

DATE-ISSUED: February 20, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Cashman; John D.	Boxborough	MA		
Riley; Paul M.	Nashua	NH		
Bahr; Raymond G.	Natick	MA		
Ye; Wei	Westford	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Cisco Technology, Inc.	San Jose	CA			02

APPL-NO: 09/ 189016 [PALM]
DATE FILED: November 9, 1998

PARENT-CASE:
RELATED APPLICATION(S) This application is a continuation of application Ser. No. 09/132,621 filed Aug. 11, 1998, which claims the benefit of U.S. Provisional Application No. 60/089,248, filed Jun. 15, 1998, the contents of which are incorporated herein by reference in their entirety.

INT-CL: [07] H02 H 3/05

US-CL-ISSUED: 714/52; 712/227
US-CL-CURRENT: 714/52; 712/227

FIELD-OF-SEARCH: 709/230, 709/236, 709/237, 709/246, 709/247, 714/18, 714/21, 714/48, 714/49, 714/50-52, 712/227

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	3678469	July 1972	Freeman et al.	340/172.5
☐	4789957	December 1988	Niehaus et al.	364/749

☐	4991133	February 1991	Davis et al.	364/900
☐	5327436	July 1994	Miyazaki	371/37.1

OTHER PUBLICATIONS

Data Sheet 7711 "Encryption Processor," Hi/fn, Inc., San Jose, CA, PRS-056 Revision 1.01 (1998).
Data Sheet 9711 "Data Compression Coprocessor," Hi/fn, Inc., San Jose, CA, PRS-0053 Revision 1.1 (Aug. 1997).

ART-UNIT: 214

PRIMARY-EXAMINER: Follansbee; John A.

ATTY-AGENT-FIRM: Hamilton, Brook, Smith & Reynolds, P.C.

ABSTRACT:

A programmable data communications device is provided to process multiple streams of data according to multiple protocols. The device is equipped with a co-processor including multiple, programmable processors allowing data to be operated on by multiple protocols. The programmable processors within the co-processor include extended instruction sets including instructions providing the operations of zero stuffing, CRC computation, partial compare, conditional move, and trie traversal. These instructions allow the processor(s) of the co-processor to more efficiently execute programs implementing data communications protocols. Since each processor is programmable, protocols standards which change may be accommodated. A network device equipped with the co-processor can handle multiple simultaneous streams of data and can implement multiple protocols on each data stream. The protocols can execute within the co-processor either independently of each other, or in conjunction with each other.

11 Claims, 16 Drawing figures

First Hit Fwd Refs

Generate Collection Print

L33: Entry 4 of 6 File: USPT Feb 20, 2001

DOCUMENT-IDENTIFIER: US 6192491 B1
TITLE: Data processor with CRC instruction set extension

Brief Summary Text (21):
According to this embodiment of the invention, the co-processor includes at least one programmable processor and at least one memory system coupled to the programmable processor. An interface is coupled to the memory system and to the programmable processor. The interface can accept at least one protocol program downloaded into the memory system from a host computer processor. The host computer can be, for example, the network device noted above. The programmable processor executes the protocol program that is downloaded. If there is more than one processor executing more than one protocol program, the co-processor can execute multiple protocols. Alternatively, if one processor exists and is supplied with multiple protocol programs, the co-processor can also execute multiple protocols. The programmable aspect of the co-processors allows it to change according to changing protocols by being re-programmed.

Brief Summary Text (22):
In another embodiment, the invention provides a multi-processor embodiment of the co-processor which can quickly process data according to the program(s) which were downloaded. According to this embodiment, first and second programmable processors are coupled to respective first and second local memory systems. A bus system is coupled to the first and second programmable processors and the first and second local memory systems. The bus system has an interface capable of being connected to a host system for transferring data between the host system and the first and second programmable processors and the first and second local memory systems.

Brief Summary Text (27):
According to another aspect of this invention, each processor circuit which serves as the first and second processor in the co-processor is itself novel. As such, the invention provides embodiments directed to a processor including an input interface for loading communications programs and communications data as well as a first unit including processor instruction logic circuits for executing first portions of the communications programs. The first unit generally corresponds to a set of instructions provided with the processor from the manufacturer of the processor.

Detailed Description Text (3):
As an example, CPU 101 in network device 104 operating on data according to this invention can configure interface 105 to accept data 109 from host computer 107 that is to be transmitted onto computer network 108. The data 109 is received and transferred into memory 102 over system bus 103. Before transmission of the data 109 to network 108, the CPU 101 instructs co-processor 100, to concurrently encrypt and compress and packetize the data 109. Once the data 109 is compressed and encrypted in packet format, co-processor 100 notifies CPU 101 of this event. The data 109 is then transmitted via interface 106 onto network 108 under the control of the CPU 101. In a similar manner, data 109 destined for host 107 is received at interface 106 from network 108 in packet form. The packets are buffered in memory 102. The data 109 is encrypted and compressed in packet form when it arrives at network device 104. CPU 101 instructs co-processor 100 to concurrently de-packetize, decompress and decrypt packets 109. When this process is complete, the

CPU 101 is notified and transfers the data 109 to host 107 via interface 105.

Detailed Description Text (19):
Signal lines 191 couple to host processor interface 181 which is a 32-bit wide synchronous, ready-controlled bus interface. Host processor interface 181 is used to program each internal programmable processor 110, 120 under the direction of CPU 101. The programming takes place over programming bus 191. The host processor interface 181 also handles the generation of interrupts to the CPU 101 during operation of co-processor 100.

Detailed Description Text (22):
Before network device 104 begins to accept and process network data 109, each processor 110, 210 in co-processor 100 must be programmed for operation (Step 702). Programs for each processor 110, 210 are downloaded into SRAM program memory 122 via the host processor interface 181 prior to the release of the reset state. Host processor interface 181 is a slave-only interface which present a memory map to CPU 101 which determines where each program is loaded.

Current US Cross Reference Classification (1):
712/227

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, April 26, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT; PLUR=NO; OP=ADJ	
☐	L33	L32 not l30	6
☐	L32	l31 and (l25 or l26)	9
☐	L31	(l12 or l13) same l28	532
☐	L30	l15 and (l25 or l26)	3
☐	L29	L27 and L28	4
☐	L28	coprocessor$1 or co-processor$1	8047
☐	L27	L24 and L25	8
☐	L26	(703/26).ccls.	281
☐	L25	(712/227).ccls.	448
☐	L24	(712/34).ccls.	217
☐	L23	condition code$1 or predicate$1	18466
☐	L22	L6 and L20	81
☐	L21	L6 same L20	2
☐	L20	(co-processor$1 or coprocessor$1)	8047
☐	L19	L6 and L14	2
☐	L18	L17 not L15	51
☐	L17	(L12 or L13) and L16	59
☐	L16	(712/35).ccls.	185
☐	L15	(L12 or L13) and L14	71
☐	L14	(712/34).ccls.	217
☐	L13	(interface$1) with program$5	80590
☐	L12	(interface$1) with configur$4	31675
☐	L11	(L8 or L9) with configur$4	6
☐	L10	(L8 or L9) with program$5	37
☐	L9	co-processor interface	79
☐	L8	coprocessor interface	226
☐	L7	coprocessor with functional units	40
☐	L6	data with L5	2654
☐	L5	L4 adj L3	24603
☐	L4	out	2555749
☐	L3	order	2415057

☐	L2	out of order	0
☐	L1	"out of order"	0

END OF SEARCH HISTORY

First Hit Fwd Refs

Generate Collection Print

L43: Entry 5 of 7 File: USPT Jul 13, 1999

US-PAT-NO: 5923893
DOCUMENT-IDENTIFIER: US 5923893 A

TITLE: Method and apparatus for interfacing a processor to a coprocessor

DATE-ISSUED: July 13, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Moyer; William C.	Dripping Springs	TX		
Arends; John	Austin	TX		
Scott; Jeffrey W.	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Motorola, Inc.	Schaumburg	IL			02

APPL-NO: 08/ 924137 [PALM]
DATE FILED: September 5, 1997

PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS The present application is related to the following U.S. patent applications: "METHOD AND APPARATUS FOR INTERFACING A PROCESSOR TO A COPROCESSOR" invented by William C. Mover et. al., having application Ser. No. 08/924,508 and Attorney Docket No. SC90634A, filed Sep. 5, 1997, and assigned to the assignee hereof; and "METHOD AND APPARATUS FOR INTERFACING A PROCESSOR TO A COPROCESSOR" invented by William C. Moyer et. al., having application Ser. No. 08/924,518 and Attorney Docket No. SC90674A, filed Sep. 5, 1997, and assigned to the assignee hereof.

INT-CL: [06] G06 F 15/76

US-CL-ISSUED: 395/800.38; 395/376, 395/309
US-CL-CURRENT: 712/38; 712/200

FIELD-OF-SEARCH: 395/800.01, 395/800.23, 395/800.31, 395/800.32, 395/800.34, 395/800.38, 395/200.38, 395/200.43, 395/280, 395/306, 395/309, 395/311, 395/376, 364/131-134

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4270167	May 1981	Koehler et al.	364/200
☐	4509116	April 1985	Lackey et al.	364/200
☐	4547849	October 1985	Louie et al.	364/200
☐	4715013	December 1987	MacGregor et al.	364/900
☐	4729094	March 1988	Zolnowsky et al.	395/800.34
☐	4731736	March 1988	Mothersole et al.	364/900
☐	4949241	August 1990	Iwasaki et al.	395/280
☐	4979102	December 1990	Tokuume	395/200.43
☐	4991078	February 1991	Wilhelm et al.	364/200
☐	5001624	March 1991	Hoffman et al.	364/200
☐	5093908	March 1992	Beacom et al.	395/376
☐	5465376	November 1995	Yoshida	395/800.34
☐	5499363	March 1996	Kaneko	395/280
☐	5504912	April 1996	Morinaga et al.	395/800.34
☐	5560029	September 1996	Papadopoulos et al.	395/800.25
☐	5577250	November 1996	Anderson et al.	395/670
☐	5603047	February 1997	Caulk, Jr.	395/800
☐	5640588	June 1997	Vegesna et al.	395/800.23
☐	5675777	October 1997	Glickman	395/561
☐	5764939	June 1998	Caulk, Jr.	395/381

OTHER PUBLICATIONS

Motorola Inc. 1990, "M68300 Family CPU32 Central Processor Unit Reference Manual", Section 7 Development Support, pp. 7-1 through 7-31.
Motorola Inc. 1995, "DSP56300 24-Bit Digital Signal Processor Family Manual", Section 10 On-Chip Emulator (OnCE.TM.), pp. 10-1 through 10-29.
Motorola Inc. 1995, "DSP56300 24-Bit Digital Signal Processor Family Manual", Section 11 JTAG (IEEE 1149.1) Test Access Port, pp. 11-1 through 11-9.

ART-UNIT: 273

PRIMARY-EXAMINER: Shah; Alpesh M.

ATTY-AGENT-FIRM: Hill; Susan C.

ABSTRACT:

A processor (12) to coprocessor (14) interface supporting multiple coprocessors (14, 16) utilizes compiler generatable software type function call and return, instruction execute, and variable load and store interface instructions. Data is moved between the processor (12) and coprocessor (14) on a bi-directional shared bus (28) either implicitly through register snooping and broadcast, or explicitly through function call and return and variable load and store interface instructions. The load and store interface instructions allow selective memory address preincrementation. The bi-directional bus (28) is potentially driven both

ways on each clock cycle. The interface separates interface instruction decode and execution. Pipelined operation is provided by indicating decoded instruction discard by negating a decode signal before an execute signal is asserted.

10 Claims, 26 Drawing figures

L43: Entry 5 of 7 File: USPT Jul 13, 1999

DOCUMENT-IDENTIFIER: US 5923893 A
TITLE: Method and apparatus for interfacing a processor to a coprocessor

Brief Summary Text (8):
It is thus important to have a coprocessor interface that is tightly coupled enough that usage of the interface is fast enough that invoking even fairly simple functions is advantageous, while abstracting the interface to such an extent that the processor architecture is isolated from as many of the details of any given coprocessor as possible. Part of this later includes making the interface programmer friendly in order to facilitate tailoring new coprocessor applications in software instead of in hardware

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L43: Entry 7 of 7 File: USPT Jan 31, 1995

US-PAT-NO: 5386503
DOCUMENT-IDENTIFIER: US 5386503 A

TITLE: Method for controlling window displays in an open systems windows environment

DATE-ISSUED: January 31, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Staggs; Kevin P.	Peoria	AZ		
Clawson; Laurence A.	Cave Creek	AZ		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Honeywell Inc.	Minneapolis	MN			02

APPL-NO: 07/ 899441 [PALM]
DATE FILED: June 16, 1992

INT-CL: [06] G06 F 15/62

US-CL-ISSUED: 395/157; 395/161, 395/200
US-CL-CURRENT: 345/788; 345/710, 345/804, 709/205

FIELD-OF-SEARCH: 395/155-161, 395/200, 395/162, 395/375, 395/325, 345/118-119, 345/120, 345/200, 345/117, 370/85.4, 370/825.52, 370/423

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4296464	October 1981	Woods et al.	395/325
☐	4556974	December 1985	Kozlik	370/85.4
☐	4607256	August 1986	Henzel	340/825.52
☐	4663619	May 1987	Staggs et al.	345/200
☐	4709347	November 1987	Kirk	395/200
☐	4962473	October 1990	Crain	364/423

☐	4974173	November 1990	Stefik et al.	395/157 X
☐	5043919	August 1991	Callaway et al.	395/162 X
☐	5050104	September 1991	Heyen et al.	395/162
☐	5226118	July 1993	Baker et al.	395/161
☐	5276801	January 1994	Heyen et al.	395/162
☐	5295244	March 1994	Deu et al.	395/161
☐	5297252	March 1994	Becker	395/160

OTHER PUBLICATIONS

Dolan et al, "X Window System Servers in Embedded Systems", COMPCON, Mar. 1990, pp. 314-319.
Treadway, "Working With Windows", Computer Graphics World, Sep. 1988, pp. 79-86.
Jones, "Introduction to the X Window System", MIT and DEC, 1989, pp. 27-64.
Sing et al, "A Critical Evaluation of PEX", IEEE Comp. Grph. and Appl., Nov. 1990, pp. 65-75.
Scheifler et al, "X Window System", MIT and DEC, 1988, pp. 1-117, 166-173, 280-285, 290-297, 306-307, 352-353, 420-429.

ART-UNIT: 233

PRIMARY-EXAMINER: Powell; Mark R.

ASSISTANT-EXAMINER: Breene; John E.

ATTY-AGENT-FIRM: Sapelli; A. A. Udseth; William Medved; A.

ABSTRACT:

The method guarantees the integrity of a process control systems display in an open system windows environment. The process control system includes an interface apparatus to at least one foreign system, and receives display information such that the display information from the foreign systems and the display information from a network of the process control system are displayed on a display unit of the process control system in a windows format in response to control information from the interface apparatus. The interface apparatus transmits the display information of the foreign systems to a display generator unit via a first input channel of the display generator unit. Control information is also transmitted to the display generator unit via the first input channel to command a display format to the display generator unit of the display unit. The display format includes a plurality of windows, one of the windows being a control view. The control view (display of the process control system) display information is transmitted to the display generator unit via a second input channel of the display generator unit. Communication checks are made between the display generator unit and the interface apparatus. If an error is detected by either unit, the first input channel is disabled, and the control view is displayed on the entire screen of the display unit and controlled by the display generator unit, thereby guaranteeing the control view is always displayed and the integrity of the process control system is maintained.

6 Claims, 10 Drawing figures

First Hit Fwd Refs

End of Result Set

Generate Collection Print

L43: Entry 7 of 7 File: USPT Jan 31, 1995

DOCUMENT-IDENTIFIER: US 5386503 A
TITLE: Method for controlling window displays in an open systems windows environment

Detailed Description Text (18):
Referring to FIG. 4, there is shown a partial functional block diagram of the existing system and the open (or opened) system. The universal operator station (US) 122 is coupled to a co-processor 200, and the co-processor is coupled to an open system, i.e., interfaces/protocols of differing design, including task control program/interface protocol (TCP/IP), open system interface (OSI), DECnet (a product of the Digital Equipment Corporation of Maynard, Mass.), . . . The universal station 122 is also connected to the LCN 120 as described above. Thus, the new universal operator station (open US) 123 includes the US 122 as described above in conjunction with the co-processor 200. The purpose of the open US 123 is to open the graphical interface to the open systems and to provide information from the closed US to the open systems. The co-processor 200 is structured to permit the interface to other systems, i.e., the open systems without jeopardizing the integrity of the existing system. The co-processor 200 of the preferred embodiment is a Motorola 68040 microprocessor which is executing the UNIX operating systems (UNIX is an operating system of the American Telephone and Telegraph Company, ATT, is readily available and is well known to those skilled in the art), and is sometimes referred to as a UNIX co-processor.

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, April 26, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT; PLUR=NO; OP=ADJ	
☐	L51	l24 and L49	0
☐	L50	l28 same L49	0
☐	L49	busy with (l12 or l13)	118
☐	L48	L45 and l24	0
☐	L47	L45 and l28	1
☐	L46	L45 same l28	0
☐	L45	L44 same (l12 or l13)	47
☐	L44	(forward Compatibil$3) or (backward compatibil$3)	1895
☐	L43	L42 same (l12 or l13)	7
☐	L42	(new or newer) with (coprocessor$1 or co-processor$1)	297
☐	L41	L40 not l32	18
☐	L40	(l12 or l13) and (l38 or l39)	18
☐	L39	old with (coprocessor$1 or co-processor$1)	26
☐	L38	older with (coprocessor$1 or co-processor$1)	8
☐	L37	obsolete with (coprocessor$1 or co-processor$1)	0
☐	L36	obsolete adj (coprocessor$1 or co-processor$1)	0
☐	L35	older adj (coprocessor$1 or co-processor$1)	0
☐	L34	old adj (coprocessor$1 or co-processor$1)	0
☐	L33	L32 not l30	6
☐	L32	l31 and (l25 or l26)	9
☐	L31	(l12 or l13) same l28	532
☐	L30	l15 and (l25 or l26)	3
☐	L29	L27 and L28	4
☐	L28	coprocessor$1 or co-processor$1	8047
☐	L27	L24 and L25	8
☐	L26	(703/26).ccls.	281
☐	L25	(712/227).ccls.	448
☐	L24	(712/34).ccls.	217
☐	L23	condition code$1 or predicate$1	18466
☐	L22	L6 and L20	81
☐	L21	L6 same L20	2

☐	L20	(co-processor$1 or coprocessor$1)	8047
☐	L19	L6 and L14	2
☐	L18	L17 not L15	51
☐	L17	(L12 or L13) and L16	59
☐	L16	(712/35).ccls.	185
☐	L15	(L12 or L13) and L14	71
☐	L14	(712/34).ccls.	217
☐	L13	(interface$1) with program$5	80590
☐	L12	(interface$1) with configur$4	31675
☐	L11	(L8 or L9) with configur$4	6
☐	L10	(L8 or L9) with program$5	37
☐	L9	co-processor interface	79
☐	L8	coprocessor interface	226
☐	L7	coprocessor with functional units	40
☐	L6	data with L5	2654
☐	L5	L4 adj L3	24603
☐	L4	out	2555749
☐	L3	order	2415057
☐	L2	out of order	0
☐	L1	"out of order"	0

END OF SEARCH HISTORY